



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

AN EXPERIMENT WITH RTEMS

by

David J. Shifflett and Thuy D. Nguyen

February 2015

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 12-02-2015		2. REPORT TYPE Technical Report		3. DATES COVERED (From-To)	
4. TITLE AND SUBTITLE AN EXPERIMENT WITH RTEMS				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) David J. Shifflett and Thuy D. Nguyen				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				8. PERFORMING ORGANIZATION REPORT NUMBER NPS-CAG-15-003	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited					
13. SUPPLEMENTARY NOTES The views expressed in this material are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.					
14. ABSTRACT The Real-Time Executive for Multiprocessor Systems (RTEMS) is an open source real-time executive used in many embedded systems. This report describes our effort to gain hands-on experience with RTEMS and provides instructions on how to build and use RTEMS in two different operating environments.					
15. SUBJECT TERMS RTEMS, SPARC simulator, Raspberry Pi.					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 43	19a. NAME OF RESPONSIBLE PERSON Thuy D. Nguyen
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (include area code) (831) 656-3989

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

**NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000**

Ronald A. Route
President

Douglas A. Hensler
Provost

The report entitled “*An Experiment with RTEMS*” was prepared for the Cyber Academic Group at the Naval Postgraduate School.

Further distribution of all or part of this report is authorized.

This report was prepared by:

David J. Shifflett
Faculty Associate – Research

Thuy D. Nguyen
Faculty Associate – Research

Reviewed by:

Released by:

Cynthia E. Irvine
Chair of Cyber Academic Group

Jeffrey D. Paduan
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Real-Time Executive for Multiprocessor Systems (RTEMS) is an open source real-time executive used in many embedded systems. This report describes our effort to gain hands-on experience with RTEMS and provides instructions on how to build and use RTEMS in two different operating environments.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION.....	11
A. MOTIVATION.....	11
B. REPORT ORGANIZATION.....	12
II. METHODS AND EQUIPMENT.....	13
A. RTEMS OVERVIEW.....	13
B. APPROACH	13
C. EQUIPMENT PREREQUISITES.....	15
1. Hardware requirements for the development system	15
2. Hardware requirements for the target system	15
3. Hardware requirements for the Raspberry Pi console	16
4. Hardware requirements for the Windows system	16
5. Software requirements for the development system.....	16
6. Software requirements for the target system	16
7. Software requirements for the Raspberry Pi console.....	16
8. Software requirements for the Windows system	17
III. PROCEDURES	19
A. DEVELOPMENT SYSTEM SET-UP.....	19
B. EXECUTION OF RTEMS IN A SIMULATOR.....	23
C. EXECUTION OF RTEMS ON RASPBERRY PI	25
IV. CONCLUSION AND FUTURE WORK.....	37
LIST OF REFERENCES	39
INITIAL DISTRIBUTION LIST	41

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Simulator experiment diagram.	14
Figure 2. Bare hardware experiment diagram.....	14
Figure 3. Listing of SPARC tools.	22
Figure 4. Compiler version for SPARC CPU.	22
Figure 5. Listing of sample applications.....	24
Figure 6. GDB simulator startup.....	25
Figure 7. Output of ticker application in the GDB simulator.	25
Figure 8. Listing of ARM tools.....	27
Figure 9. Compiler version for ARM CPU.....	27
Figure 10. Installed Raspberry Pi tools.....	28
Figure 11. Raspberry Pi sample executables.	29
Figure 12. RKI build status.	30
Figure 13. A Raspberry Pi bootable file	30
Figure 14. Contents of SD card ready for booting on the Raspberry Pi	32
Figure 15. Output of ticker application on Raspberry Pi	33
Figure 16. RKI start-up screen.....	34
Figure 17. RKI file-related commands	35
Figure 18. RKI whetstone results.....	36
Figure 19. RKI resource usage.....	36

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This report describes our effort to gain hands-on experience with the Real-Time Executive for Multiprocessor Systems (RTEMS) in support of our investigation of network covert communications in space systems--specifically in the context of commercially hosted payloads.

A. MOTIVATION

RTEMS is an open source, high performance, real-time executive used in many embedded systems, including space flight and aviation [1]. RTEMS supports a number of space-qualified microprocessors such as LEON (SPARC) [2] and RAD750 (PowerPC) [3], making it a popular candidate for space systems. Space missions that utilize RTEMS for on-board instruments include four spacecraft developed by the European Space Agency (ESA)—Herschel, Planck, Pleiades and Aeolus, and the Mars Reconnaissance Orbiter (National Aeronautics and Space Administration). On the ESA satellites, RTEMS is used in the scheduling engine of the Spacecraft Management Unit [4]. For the Mars orbiter, RTEMS is used in Electra Proximity Link software-defined radio which is responsible for relaying commands and data between Earth and landers on Mars [5]. RTEMS is also one of the real-time operating systems that NASA's Core Flight Software (CFS) supports; CFS is used on the Lunar Reconnaissance Orbiter and other NASA missions [6].

The rising cost of developing and maintaining government-owned space vehicles has pushed the Department of Defense to pursue a piggyback approach that allows hosting government-supplied payloads on commercial space platforms. These commercially hosted payloads require stringent confidentiality protection to protect against illegal information leakage. We are currently investigating covert channel attacks on network protocols used in spacecraft that host multiple payloads operating at different sensitivity levels, i.e., spacecraft with multilevel security capabilities. We have identified several potential covert channels in the MIL-STD-1553B and SpaceWire protocols [7] and, since RTEMS supports these protocols, knowing the inner working of RTEMS will

allow further experimentation to determine real-world exploitation scenarios and defenses against the identified covert channels.

B. REPORT ORGANIZATION

The remainder of this report begins with information about the RTEMS distribution and the approach we used for this experiment. Next, we provide instructions on how to build and use RTEMS in two different operating environments. Last, we close with a brief description of future work.

II. METHODS AND EQUIPMENT

Our experimentation with RTEMS includes two scenarios: running RTEMS in a hardware simulator and running RTEMS on a single board computer (SBC). This section discusses the choices made for each of these experiments, and the equipment necessary to perform the experiments.

A. RTEMS OVERVIEW

“The Real-Time Executive for Multiprocessor Systems or RTEMS is a [sic] open source fully featured Real Time Operating System or RTOS that supports a variety of open standard application programming interfaces (API) and interface standards such as POSIX and BSD sockets” [8]. RTEMS is available on a wide variety of CPUs (e.g., ARM, SPARC, PowerPC and Intel) [9] and an even wider range of processor boards [10]. Support for a processor board is provided through a Board Support Package (BSP). To use RTEMS, a developer first decides the target CPU and BSP, builds the RTEMS tools (e.g., compiler, linker, and debugger) for that combination, and then uses those tools to build RTEMS and the desired RTEMS application. The result is a single executable file containing both RTEMS and the application.

RTEMS provides standard operating system interfaces for file systems, graphics libraries, networking, memory management, task management, etc. Many of these interfaces are POSIX-compliant for portability.

B. APPROACH

Our strategy consists of two experiments. The first experiment tests RTEMS on a simulator that is distributed with RTEMS. Using the simulator allows us to verify the build environment, experiment with RTEMS functionalities, and prepare for running it on an SBC. The second experiment exercises RTEMS on bare hardware using a selected SBC. Figure 1 shows a block diagram of the first experiment and Figure 2 shows a diagram of the second experiment.

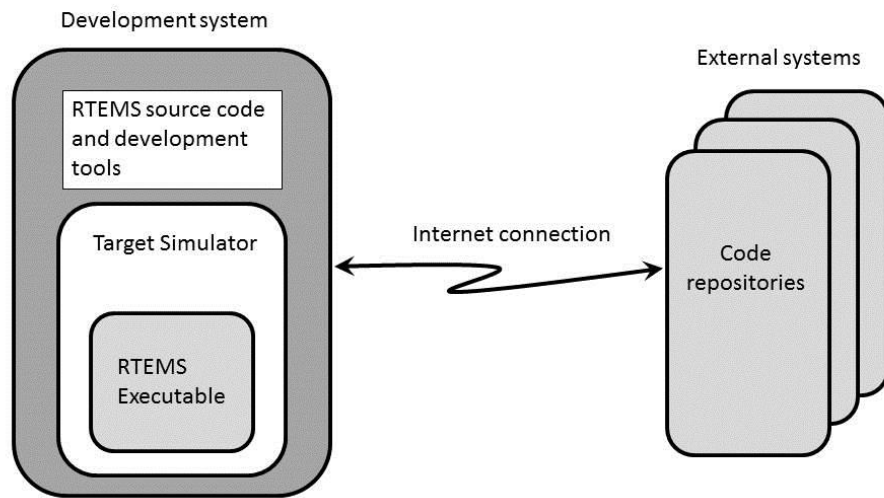


Figure 1. Simulator experiment diagram.

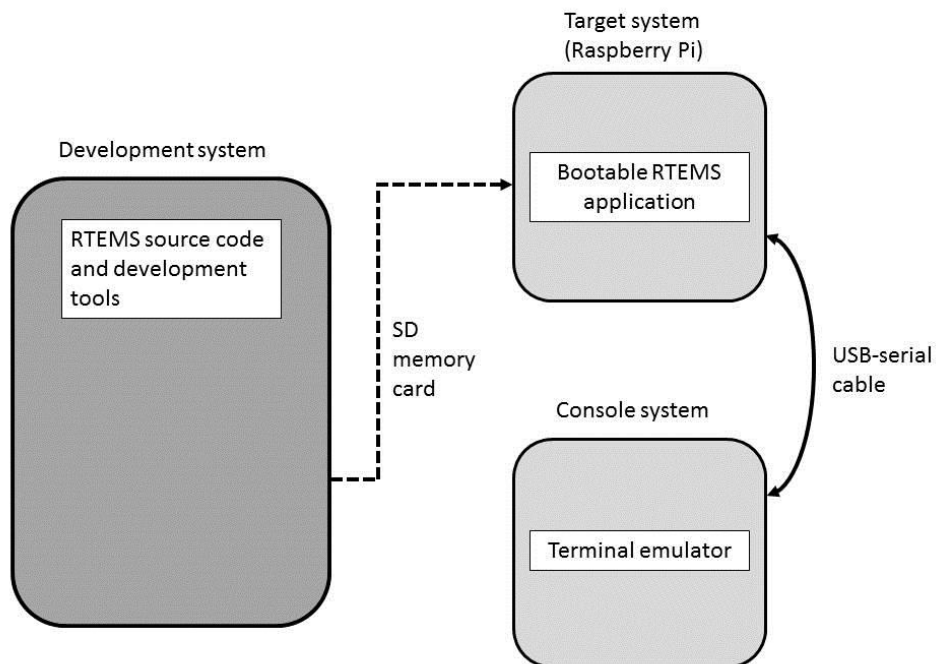


Figure 2. Bare hardware experiment diagram.

For the bare hardware experiment, the first step is to choose an SBC on which RTEMS can run. Support for MIL-STD-1553B [11] and SpaceWire [12], and compatibility with RTEMS, through an existing BSP, are the deciding factors.

The candidate boards were: BAE RAD750 [13], Motorola MCP750 [25], Raspberry Pi [16], and Aeroflex-Gaisler GR-LEON4-ITX [14] and GR-CPCI-LEON4-N2X [15]. Four of these boards were rejected for two reasons: 1) availability—the MCP750 is no longer produced, though used versions are available and 2) cost—the RAD750 and LEON4 boards are too expensive for research experimentation.

The Raspberry Pi was used for the bare hardware experiment for two reasons: the existence of an RTEMS BSP for the Raspberry Pi and the availability of online resources for building RTEMS to run on the Raspberry Pi [17].

C. EQUIPMENT PREREQUISITES

This section describes the hardware and software requirements.

1. Hardware requirements for the development system

For both experiments, a development system meeting the following requirements is required:

- The system must have at least a 100 GB hard disk and 4 GB of RAM.
- The system must have Internet connectivity for updates, package installs, and access to the *git* repositories.

The development system can be either a physical computer or a virtual machine.

2. Hardware requirements for the target system

The simulator experiment runs directly on the development system.

The bare hardware experiment runs on a Raspberry Pi model B+ board. No special configuration of the Raspberry Pi is required for this experiment.

A micro Secure Digital (SD) memory card and an SD card reader are required to copy files from the development system to the SD card for use with the Raspberry Pi. We used the Insignia SD/MMC Memory Card Reader (model NS-CR2021).

3. **Hardware requirements for the Raspberry Pi console**

A separate system is required to act as a console for the Raspberry Pi. This system must be able to establish a serial connection via a USB interface, usually through a terminal emulation program, such as Putty [26] or CoolTerm [27]. We used an Apple Mac Pro running the CoolTerm terminal emulator software. This system is connected to the Raspberry Pi via a special USB-serial cable [20].

4. **Hardware requirements for the Windows system**

We used a Windows system to copy Raspberry Pi boot files, and RTEMS executable files, to an SD card for booting the Raspberry Pi.

5. **Software requirements for the development system**

The development system needs the base Fedora 19 installation [23] and the following additional packages required by RTEMS: *ncurses-devel*, *git*, *bison*, *gcc*, *cvs*, *gcc-c++*, *flex*, *texinfo*, *patch*, *perl-Text-ParseWords*, *zlib-devel*, and *python-devel*.

The following software must be downloaded from *git* repositories as needed during the experimentation procedures:

- RTEMS Source Builder
- RTEMS source
- An example RTEMS application for the Raspberry Pi

6. **Software requirements for the target system**

The software needed to boot the Raspberry Pi can be downloaded from the Raspberry Pi *git* repository (<http://github.com/raspberrypi/firmware/tree/master/boot>). The files needed are:

- *bootcode.bin*
- *start.elf*

7. **Software requirements for the Raspberry Pi console**

The Raspberry Pi console system needs a terminal emulator program, such as Putty or CoolTerm.

8. **Software requirements for the Windows system**

The Windows system needs the drivers for the SD card reader.

THIS PAGE INTENTIONALLY LEFT BLANK

III. PROCEDURES

This section describes the procedures to run an RTEMS executable in a SPARC simulator and on a Raspberry Pi.

Section A provides instructions for setting up a development system to handle the building of RTEMS and RTEMS applications. Section B gives instructions for building and running an RTEMS executable within a simulator—an RTEMS executable includes both RTEMS and an application. Finally, Section C contains instructions for how to build an RTEMS executable for the Raspberry Pi and run it on a Raspberry Pi.

In the instructions below the reader will enter many commands in a terminal window on the development system. These commands are formatted in `Courier 12 font`.

A. DEVELOPMENT SYSTEM SET-UP

In this example Fedora 19 was chosen to be the host operating system because it is well understood and there is documentation for installing RTEMS on Fedora 19 [18]. Preparation of this system includes installing software packages needed by RTEMS, using the RTEMS Source Builder (RSB) to build the tools (compiler, linker, debugger, etc.) required to build RTEMS, making a local copy of the RTEMS source repository, and building the RTEMS BSP for the target system.

Step 1: Install Fedora 19

During the installation of Fedora 19, a root user and a development user must be created. These procedures assume the development user is named *user* and the privileged root user is named *root*. The full details of installing Fedora 19 are beyond the scope of this report, but can be found in [23].

Step 2: Enable execution of privileged commands

Privileged commands must be invoked during the preparation of the development system, and installation of RTEMS. The steps to enable the development user to run privileged commands are:

1. Login as *user*.
2. Start a terminal window.
3. Switch privileges to run as *root*. Execute `'su -'` and enter the password for *root* when prompted.
4. Enable *user* to run privileged commands. At the root prompt, execute `'visudo'`. Directions for using this editor can be found in [28].
5. Scroll down to the line that looks like `'root ALL=(ALL) ALL'`.
6. Copy and paste this line immediately below the line being copied.
7. In the new line, change *root* to *user*.
8. Save the file and exit the editor.
9. Exit from the root prompt. Execute `'exit'`.

Step 3: Update Fedora 19

RTEMS needs software beyond the base installation of Fedora 19. The steps to update the system and add packages needed by RTEMS are:

1. Login as *user* and start a terminal window.
2. Completely update the base operating system. Execute `'sudo yum update'`.
3. Add the packages needed by RTEMS. Execute `'sudo yum install ncurses-devel git bison gcc cvs gcc-c++ flex texinfo patch perl-Text-ParseWords zlib-devel python-devel'`.

Step 4: Build RTEMS tools

The RTEMS tools are built using the RTEMS Source Builder. The steps to do this are:

1. Login as *user* and execute the following commands.
2. `mkdir -p ~/development/rtems/src`
3. `cd -p ~/development/rtems/src`
4. `git clone git://git.rtems.org/rtems-source-builder.git`
5. `cd rtems-source-builder/`

6. `source-builder/sb-check`

This command verifies that all necessary dependencies have been satisfied. If any errors are shown, or dependencies are not satisfied, resolve the problems, or missing dependencies, before continuing.

7. `cd rtems`

8. `../source-builder/sb-set-builder --list-bsets`

This command lists available versions and architectures.

9. `../source-builder/sb-set-builder --log=l-sparc.txt -
-prefix=$HOME/development/rtems/4.11 --with-rtems
4.11/rtems-sparc`

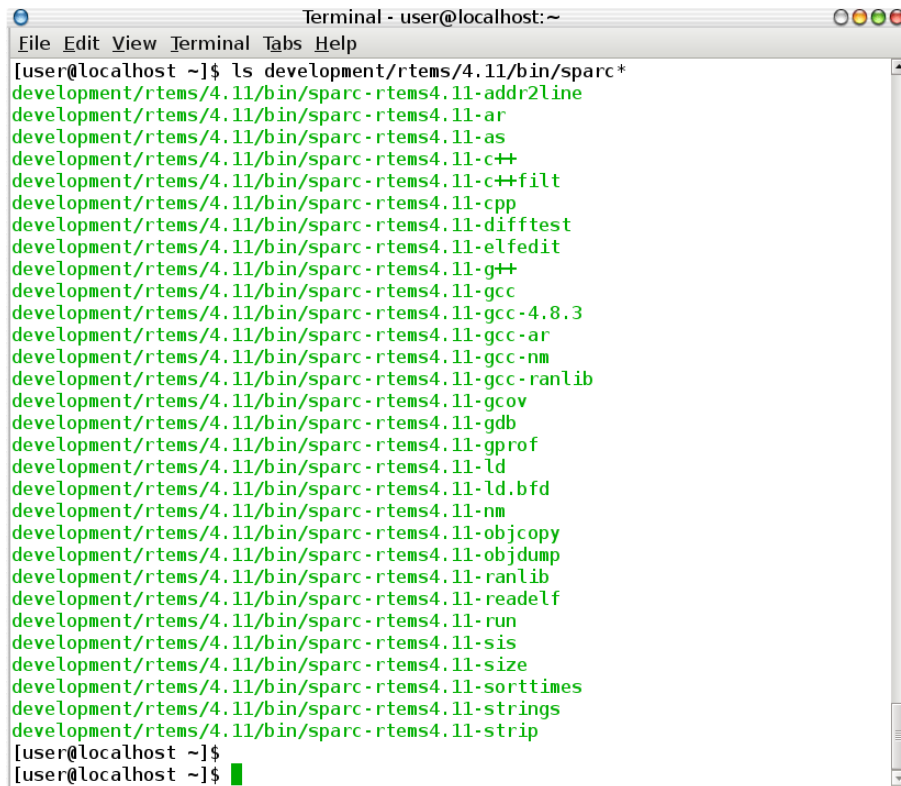
- As specified by the last parameter, this command builds the RTEMS tools version 4.11 for the SPARC CPU. Other version and CPU combinations are available as listed by the output of step 8 above.
- This command should build completely without errors. The last line output should look similar to: '*Build Set Time: 2:08:47.078118*'.
- The file specified by the `--log` parameter will contain all build commands and output.

10. `ls ~/development/rtems/4.11/bin/sparc*`

This command lists the tools created.

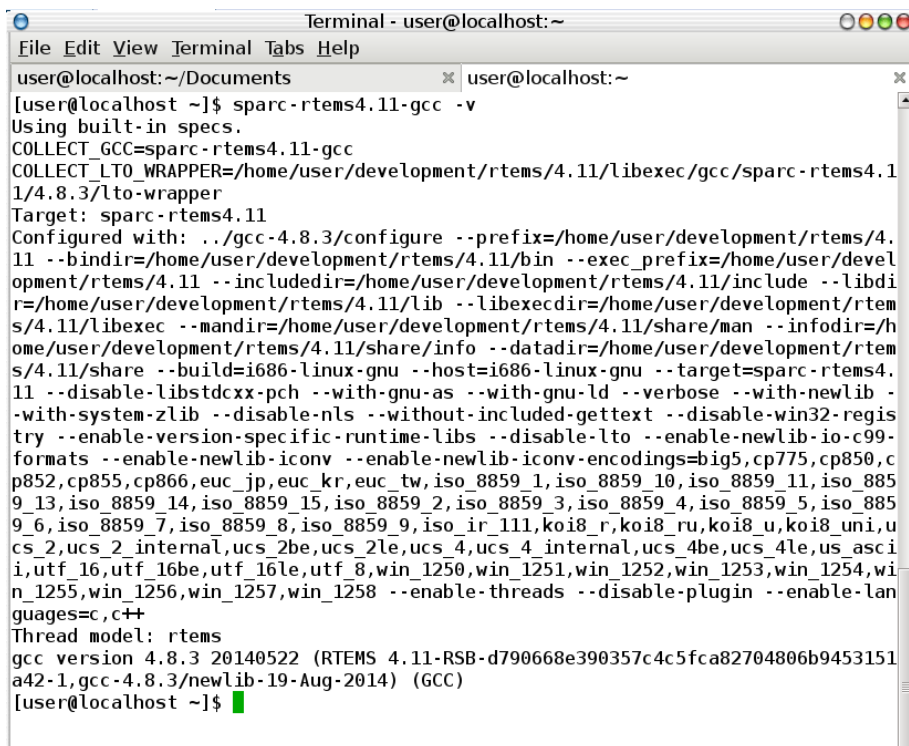
The resulting list should be similar to that shown in Figure 3.

11. To verify the tools are built successfully, check the version of a particular tool. For example, to check the version of the file *sparc-rtems4.11-gcc*, execute '`sparc-rtems4.11-gcc -v`'. A result similar to that shown in Figure 4 is expected.



```
Terminal - user@localhost: ~
File Edit View Terminal Tabs Help
[user@localhost ~]$ ls development/rtems/4.11/bin/sparc*
development/rtems/4.11/bin/sparc-rtems4.11-addr2line
development/rtems/4.11/bin/sparc-rtems4.11-ar
development/rtems/4.11/bin/sparc-rtems4.11-as
development/rtems/4.11/bin/sparc-rtems4.11-c++
development/rtems/4.11/bin/sparc-rtems4.11-c++filt
development/rtems/4.11/bin/sparc-rtems4.11-cpp
development/rtems/4.11/bin/sparc-rtems4.11-difftest
development/rtems/4.11/bin/sparc-rtems4.11-elfedit
development/rtems/4.11/bin/sparc-rtems4.11-g++
development/rtems/4.11/bin/sparc-rtems4.11-gcc
development/rtems/4.11/bin/sparc-rtems4.11-gcc-4.8.3
development/rtems/4.11/bin/sparc-rtems4.11-gcc-ar
development/rtems/4.11/bin/sparc-rtems4.11-gcc-nm
development/rtems/4.11/bin/sparc-rtems4.11-gcc-ranlib
development/rtems/4.11/bin/sparc-rtems4.11-gcov
development/rtems/4.11/bin/sparc-rtems4.11-gdb
development/rtems/4.11/bin/sparc-rtems4.11-gprof
development/rtems/4.11/bin/sparc-rtems4.11-ld
development/rtems/4.11/bin/sparc-rtems4.11-ld.bfd
development/rtems/4.11/bin/sparc-rtems4.11-nm
development/rtems/4.11/bin/sparc-rtems4.11-objcopy
development/rtems/4.11/bin/sparc-rtems4.11-objdump
development/rtems/4.11/bin/sparc-rtems4.11-ranlib
development/rtems/4.11/bin/sparc-rtems4.11-readelf
development/rtems/4.11/bin/sparc-rtems4.11-run
development/rtems/4.11/bin/sparc-rtems4.11-sis
development/rtems/4.11/bin/sparc-rtems4.11-size
development/rtems/4.11/bin/sparc-rtems4.11-sorttimes
development/rtems/4.11/bin/sparc-rtems4.11-strings
development/rtems/4.11/bin/sparc-rtems4.11-strip
[user@localhost ~]$
[user@localhost ~]$
```

Figure 3. Listing of SPARC tools.



```
Terminal - user@localhost: ~
File Edit View Terminal Tabs Help
user@localhost:~/Documents x user@localhost: ~ x
[user@localhost ~]$ sparc-rtems4.11-gcc -v
Using built-in specs.
COLLECT_GCC=sparc-rtems4.11-gcc
COLLECT_LTO_WRAPPER=/home/user/development/rtems/4.11/libexec/gcc/sparc-rtems4.11/4.8.3/lto-wrapper
Target: sparc-rtems4.11
Configured with: ../gcc-4.8.3/configure --prefix=/home/user/development/rtems/4.11 --bindir=/home/user/development/rtems/4.11/bin --exec_prefix=/home/user/development/rtems/4.11 --includedir=/home/user/development/rtems/4.11/include --libdir=/home/user/development/rtems/4.11/lib --libexecdir=/home/user/development/rtems/4.11/libexec --mandir=/home/user/development/rtems/4.11/share/man --infodir=/home/user/development/rtems/4.11/share/info --datadir=/home/user/development/rtems/4.11/share --build=i686-linux-gnu --host=i686-linux-gnu --target=sparc-rtems4.11 --disable-libstdcxx-pch --with-gnu-as --with-gnu-ld --verbose --with-newlib --with-system-zlib --disable-nls --without-included-gettext --disable-win32-registry --enable-version-specific-runtime-libs --disable-lto --enable-newlib-io-c99-formats --enable-newlib-iconv --enable-newlib-iconv-encodings=big5,cp775,cp850,cp852,cp855,cp866,euc_jp,euc_kr,euc_tw,iso_8859_1,iso_8859_10,iso_8859_11,iso_8859_13,iso_8859_14,iso_8859_15,iso_8859_2,iso_8859_3,iso_8859_4,iso_8859_5,iso_8859_6,iso_8859_7,iso_8859_8,iso_8859_9,iso_ir_111,koi8_r,koi8_ru,koi8_u,koi8_uni,ucs_2,ucs_2_internal,ucs_2be,ucs_2le,ucs_4,ucs_4_internal,ucs_4be,ucs_4le,us_ascii,utf_16,utf_16be,utf_16le,utf_8,win_1250,win_1251,win_1252,win_1253,win_1254,win_1255,win_1256,win_1257,win_1258 --enable-threads --disable-plugin --enable-languages=c,c++
Thread model: rtems
gcc version 4.8.3 20140522 (RTEMS 4.11-RSB-d790668e390357c4c5fca82704806b9453151a42-1,gcc-4.8.3/newlib-19-Aug-2014) (GCC)
[user@localhost ~]$
```

Figure 4. Compiler version for SPARC CPU.

B. EXECUTION OF RTEMS IN A SIMULATOR

The ability to execute RTEMS in a simulator allows quick verification of the build environment, and quick experimentation with changes to RTEMS or the application.

Step 1: Build RTEMS and sample applications

First obtain the RTEMS source from the RTEMS *git* repository, then prepare RTEMS for building, finally configure, and build RTEMS. The result is a set of RTEMS executables that are built for the SPARC Instruction Simulator (*'sis'*) BSP. This BSP allows the generated executables to run within the SPARC simulator for testing, verification and experimentation. The steps to build RTEMS are (see [19]):

1. Login as *user* and execute the following commands.
2. `cd ~/development/rtems/src/`
3. `git clone git://git.rtems.org/rtems.git rtems`
4. `export PATH=$HOME/development/rtems/4.11/bin:$PATH`
5. `cd rtems`
6. `./bootstrap`
7. `cd ..`
8. `mkdir b-sis`
9. `cd b-sis`
10. `../rtems/configure --target=sparc-rtems4.11 --
enable-rtemsbsp=sis --enable-tests=samples --
disable-posix`
11. `make`
12. `sudo PATH=$HOME/development/rtems/4.11/bin:$PATH
make install`
13. To verify that the sample applications were built, execute `'find . -name
'*.exe''`. A result similar to that shown in Figure 5 is expected.

A terminal window titled "Terminal - user@localhost: ~/development/rtems/src/b-sis". The window shows the output of the command `find . -name '*.exe'`. The output lists ten sample applications: `./sparc-rtems4.11/c/sis/testsuites/samples/base_sp/base_sp.exe`, `./sparc-rtems4.11/c/sis/testsuites/samples/minimum/minimum.exe`, `./sparc-rtems4.11/c/sis/testsuites/samples/unlimited/unlimited.exe`, `./sparc-rtems4.11/c/sis/testsuites/samples/fileio/fileio.exe`, `./sparc-rtems4.11/c/sis/testsuites/samples/pppd/pppd.exe`, `./sparc-rtems4.11/c/sis/testsuites/samples/loopback/loopback.exe`, `./sparc-rtems4.11/c/sis/testsuites/samples/paranoia/paranoia.exe`, `./sparc-rtems4.11/c/sis/testsuites/samples/nsecs/nsecs.exe`, `./sparc-rtems4.11/c/sis/testsuites/samples/capture/capture.exe`, and `./sparc-rtems4.11/c/sis/testsuites/samples/hello/hello.exe`. The terminal prompt is `[user@localhost b-sis]$`.

```
Terminal - user@localhost: ~/development/rtems/src/b-sis
File Edit View Terminal Tabs Help
[user@localhost b-sis]$ find . -name '*.exe'
./sparc-rtems4.11/c/sis/testsuites/samples/base_sp/base_sp.exe
./sparc-rtems4.11/c/sis/testsuites/samples/minimum/minimum.exe
./sparc-rtems4.11/c/sis/testsuites/samples/unlimited/unlimited.exe
./sparc-rtems4.11/c/sis/testsuites/samples/fileio/fileio.exe
./sparc-rtems4.11/c/sis/testsuites/samples/pppd/pppd.exe
./sparc-rtems4.11/c/sis/testsuites/samples/loopback/loopback.exe
./sparc-rtems4.11/c/sis/testsuites/samples/paranoia/paranoia.exe
./sparc-rtems4.11/c/sis/testsuites/samples/nsecs/nsecs.exe
./sparc-rtems4.11/c/sis/testsuites/samples/capture/capture.exe
./sparc-rtems4.11/c/sis/testsuites/samples/hello/hello.exe
./sparc-rtems4.11/c/sis/testsuites/samples/ticker/ticker.exe
[user@localhost b-sis]$
```

Figure 5. Listing of sample applications.

Step 2: Start the simulator

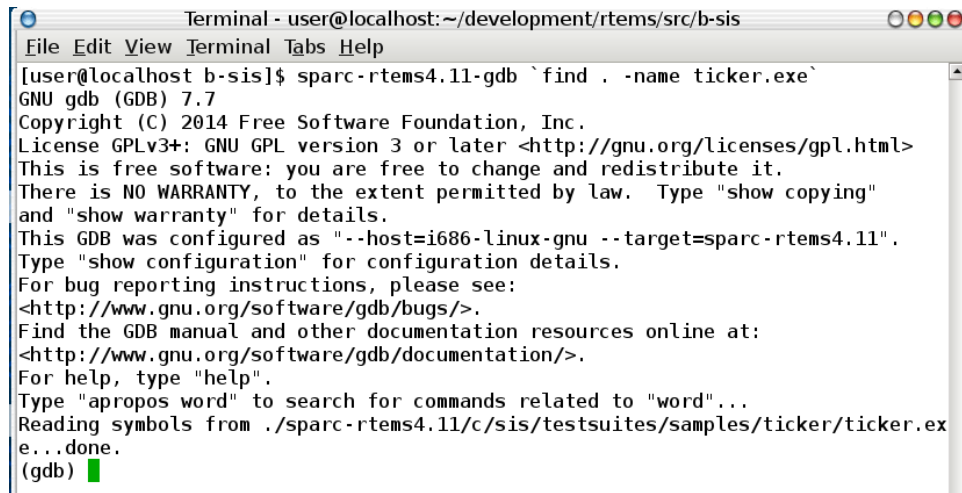
The simulator is a feature built into the debugger (GDB). The steps to start the debugger with a target program are:

1. Login as *user* and execute the following commands.
2. `export PATH=$HOME/development/rtems/4.11/bin:$PATH`
3. `cd ~/development/rtems/src/b-sis`
4. `sparc-rtems4.11-gdb `find . -name ticker.exe``
This command starts the simulator with the *ticker* application as the target program. A result similar to that shown in Figure 6 is expected. Note: the command above uses the grave, or backtick, character, not an apostrophe, or single quote, character.

Step 3: Run the sample application

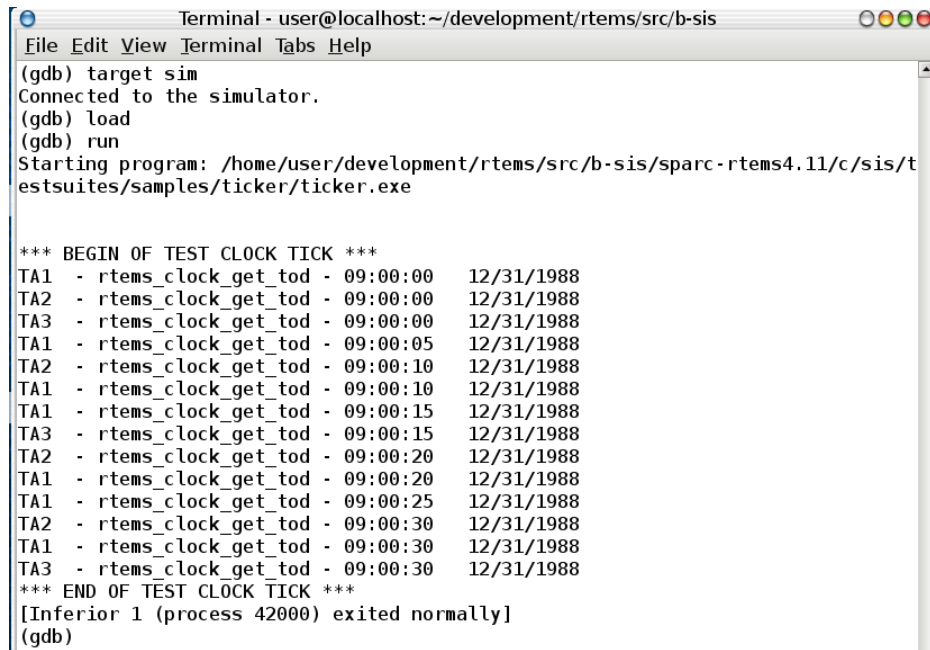
From the debugger prompt, load and run the target application. The steps to run the *ticker* application are:

1. `target sim`
2. `load`
3. `run`
4. `quit`
5. A result similar to that shown in **Error! Reference source not found.**Figure 7 is expected.

A terminal window titled "Terminal - user@localhost: ~/development/rtems/src/b-sis" showing the startup of the GDB simulator. The user runs the command "sparc-rtems4.11-gdb `find . -name ticker.exe`". The output shows the GNU gdb (GDB) 7.7 version, copyright information, and the configuration for the sparc-rtems4.11 target. The prompt is "(gdb)".

```
Terminal - user@localhost: ~/development/rtems/src/b-sis
File Edit View Terminal Tabs Help
[user@localhost b-sis]$ sparc-rtems4.11-gdb `find . -name ticker.exe`
GNU gdb (GDB) 7.7
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-linux-gnu --target=sparc-rtems4.11".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./sparc-rtems4.11/c/sis/testsuites/samples/ticker/ticker.ex
e...done.
(gdb) █
```

Figure 6. GDB simulator startup.

A terminal window titled "Terminal - user@localhost: ~/development/rtems/src/b-sis" showing the output of the ticker application in the GDB simulator. The user enters commands "target sim", "load", and "run". The output shows the program starting, followed by a table of test results for the "rtems_clock_get_tod" function. The prompt is "(gdb)".

```
Terminal - user@localhost: ~/development/rtems/src/b-sis
File Edit View Terminal Tabs Help
(gdb) target sim
Connected to the simulator.
(gdb) load
(gdb) run
Starting program: /home/user/development/rtems/src/b-sis/sparc-rtems4.11/c/sis/t
estsuites/samples/ticker/ticker.exe

*** BEGIN OF TEST CLOCK TICK ***
TA1 - rtems_clock_get_tod - 09:00:00 12/31/1988
TA2 - rtems_clock_get_tod - 09:00:00 12/31/1988
TA3 - rtems_clock_get_tod - 09:00:00 12/31/1988
TA1 - rtems_clock_get_tod - 09:00:05 12/31/1988
TA2 - rtems_clock_get_tod - 09:00:10 12/31/1988
TA1 - rtems_clock_get_tod - 09:00:10 12/31/1988
TA1 - rtems_clock_get_tod - 09:00:15 12/31/1988
TA3 - rtems_clock_get_tod - 09:00:15 12/31/1988
TA2 - rtems_clock_get_tod - 09:00:20 12/31/1988
TA1 - rtems_clock_get_tod - 09:00:20 12/31/1988
TA1 - rtems_clock_get_tod - 09:00:25 12/31/1988
TA2 - rtems_clock_get_tod - 09:00:30 12/31/1988
TA1 - rtems_clock_get_tod - 09:00:30 12/31/1988
TA3 - rtems_clock_get_tod - 09:00:30 12/31/1988
*** END OF TEST CLOCK TICK ***
[Inferior 1 (process 42000) exited normally]
(gdb) ..
```

Figure 7. Output of ticker application in the GDB simulator.

C. EXECUTION OF RTEMS ON RASPBERRY PI

To run an RTEMS executable on the Raspberry Pi platform, it is assumed that the following dependencies are satisfied by performing the steps in sections A and B:

1. The RTEMS source builder has been installed in
\$HOME/development/rtems/src/rtems-source-builder.
2. The RTEMS source has been installed in \$HOME/development/rtems/src.

The steps described in this section closely follow those in sections A and B, i.e., build the RTEMS tools (for the ARM CPU), build the RTEMS BSP (for the Raspberry Pi), build the executables, and run the executable (see [17]).

This experiment includes running two executables on the Raspberry Pi. The first executable is the RTEMS sample *ticker* application and it is used to verify the build environment, and the configuration of the hardware and console system. The second executable is a more complicated RTEMS application that allows exploration of the running RTEMS system.

Step 1: Build RTEMS tools

The steps to build the RTEMS tools for version 4.11 and the ARM CPU are:

1. Login as *user* and execute the following commands.
2. `cd ~/development/rtems/src/rtems-source-builder/rtems`

This command sets the current directory to the RTEMS source builder directory.

3. `../source-builder/sb-set-builder --log=l-arm.txt --prefix=$HOME/development/rtems/4.11 4.11/rtems-arm`

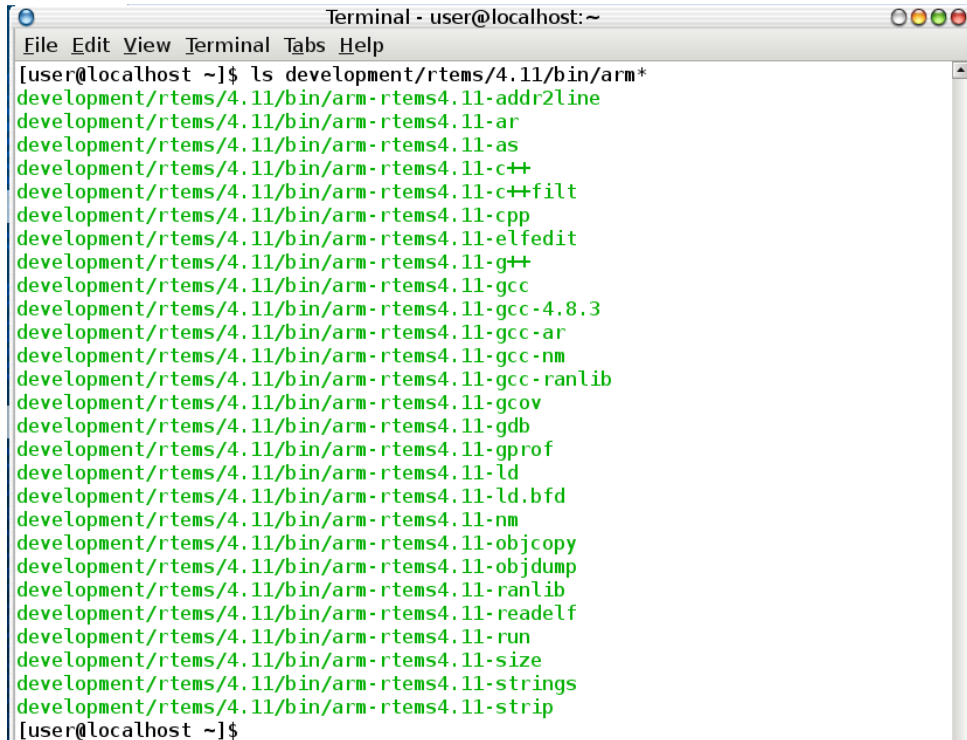
This command builds the RTEMS tools for version 4.11 for the ARM CPU.

4. `ls ~/development/rtems/4.11/bin/arm*`

The resulting list should be similar to that shown in Figure 8.

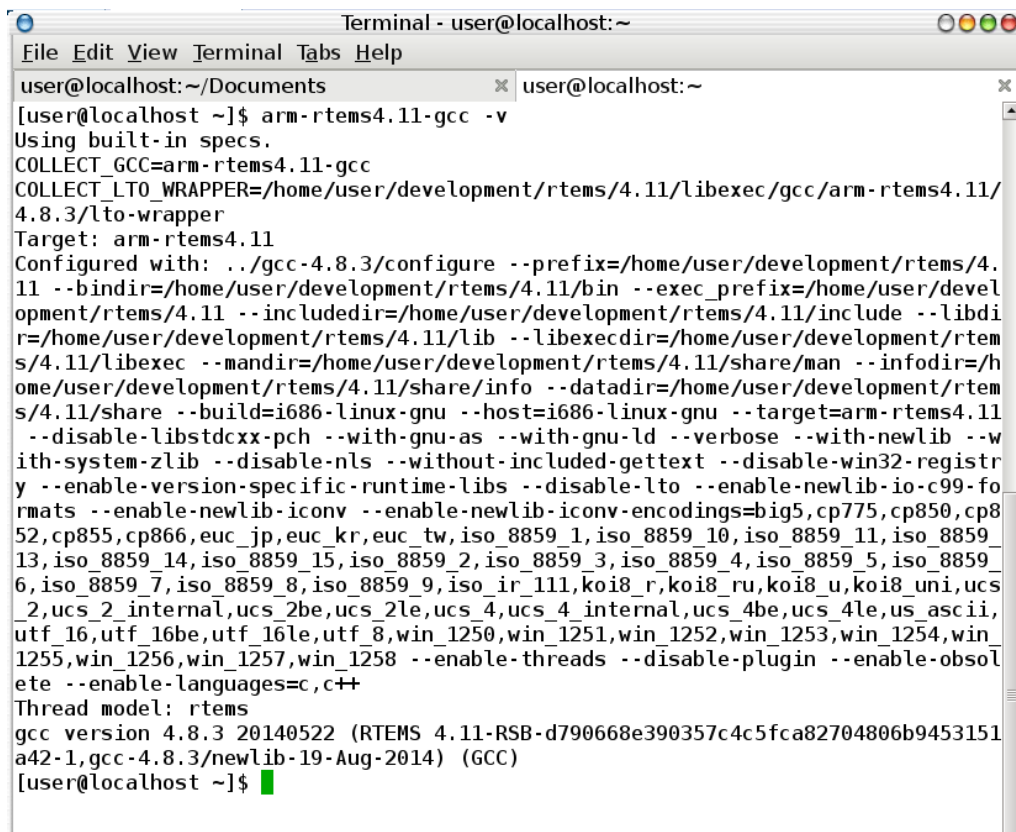
5. To verify the tools are built successfully, check the version of a particular tool. For example, to check the version of the file *arm-rtems4.11-gcc*, execute `'arm-rtems4.11-gcc -v'`. A result similar to that shown in **Error!**

Reference source not found.Figure 9 is expected.



```
Terminal - user@localhost:~
File Edit View Terminal Tabs Help
[user@localhost ~]$ ls development/rtems/4.11/bin/arm*
development/rtems/4.11/bin/arm-rtems4.11-addr2line
development/rtems/4.11/bin/arm-rtems4.11-ar
development/rtems/4.11/bin/arm-rtems4.11-as
development/rtems/4.11/bin/arm-rtems4.11-c++
development/rtems/4.11/bin/arm-rtems4.11-c++filt
development/rtems/4.11/bin/arm-rtems4.11-cpp
development/rtems/4.11/bin/arm-rtems4.11-elfedit
development/rtems/4.11/bin/arm-rtems4.11-g++
development/rtems/4.11/bin/arm-rtems4.11-gcc
development/rtems/4.11/bin/arm-rtems4.11-gcc-4.8.3
development/rtems/4.11/bin/arm-rtems4.11-gcc-ar
development/rtems/4.11/bin/arm-rtems4.11-gcc-nm
development/rtems/4.11/bin/arm-rtems4.11-gcc-ranlib
development/rtems/4.11/bin/arm-rtems4.11-gcov
development/rtems/4.11/bin/arm-rtems4.11-gdb
development/rtems/4.11/bin/arm-rtems4.11-gprof
development/rtems/4.11/bin/arm-rtems4.11-ld
development/rtems/4.11/bin/arm-rtems4.11-ld.bfd
development/rtems/4.11/bin/arm-rtems4.11-nm
development/rtems/4.11/bin/arm-rtems4.11-objcopy
development/rtems/4.11/bin/arm-rtems4.11-objdump
development/rtems/4.11/bin/arm-rtems4.11-ranlib
development/rtems/4.11/bin/arm-rtems4.11-readelf
development/rtems/4.11/bin/arm-rtems4.11-run
development/rtems/4.11/bin/arm-rtems4.11-size
development/rtems/4.11/bin/arm-rtems4.11-strings
development/rtems/4.11/bin/arm-rtems4.11-strip
[user@localhost ~]$
```

Figure 8. Listing of ARM tools.



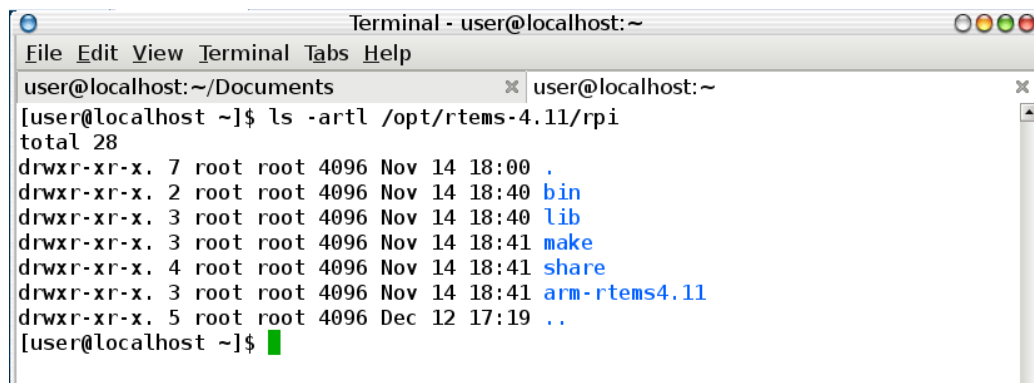
```
Terminal - user@localhost:~
File Edit View Terminal Tabs Help
user@localhost:~/Documents x user@localhost:~ x
[user@localhost ~]$ arm-rtems4.11-gcc -v
Using built-in specs.
COLLECT_GCC=arm-rtems4.11-gcc
COLLECT_LTO_WRAPPER=/home/user/development/rtems/4.11/libexec/gcc/arm-rtems4.11/4.8.3/lto-wrapper
Target: arm-rtems4.11
Configured with: ../gcc-4.8.3/configure --prefix=/home/user/development/rtems/4.11 --bindir=/home/user/development/rtems/4.11/bin --exec_prefix=/home/user/development/rtems/4.11 --includedir=/home/user/development/rtems/4.11/include --libdir=/home/user/development/rtems/4.11/lib --libexecdir=/home/user/development/rtems/4.11/libexec --mandir=/home/user/development/rtems/4.11/share/man --infodir=/home/user/development/rtems/4.11/share/info --datadir=/home/user/development/rtems/4.11/share --build=i686-linux-gnu --host=i686-linux-gnu --target=arm-rtems4.11 --disable-libstdcxx-pch --with-gnu-as --with-gnu-ld --verbose --with-newlib --with-system-zlib --disable-nls --without-included-gettext --disable-win32-registry --enable-version-specific-runtime-libs --disable-lto --enable-newlib-io-c99-formats --enable-newlib-iconv --enable-newlib-iconv-encodings=big5,cp775,cp850,cp852,cp855,cp866,euc_jp,euc_kr,euc_tw,iso_8859_1,iso_8859_10,iso_8859_11,iso_8859_13,iso_8859_14,iso_8859_15,iso_8859_2,iso_8859_3,iso_8859_4,iso_8859_5,iso_8859_6,iso_8859_7,iso_8859_8,iso_8859_9,iso_ir_111,koi8_r,koi8_ru,koi8_u,koi8_uni,ucs_2,ucs_2_internal,ucs_2be,ucs_2le,ucs_4,ucs_4_internal,ucs_4be,ucs_4le,us_ascii,utf_16,utf_16be,utf_16le,utf_8,win_1250,win_1251,win_1252,win_1253,win_1254,win_1255,win_1256,win_1257,win_1258 --enable-plugin --enable-obsolete --enable-languages=c,c++
Thread model: rtems
gcc version 4.8.3 20140522 (RTEMS 4.11-RSB-d790668e390357c4c5fca82704806b9453151a42-1,gcc-4.8.3/newlib-19-Aug-2014) (GCC)
[user@localhost ~]$
```

Figure 9. Compiler version for ARM CPU.

Step 2: Build RTEMS and sample executables

The steps to build RTEMS and a set of sample executables for the Raspberry Pi are:

1. Login as *user* and execute the following commands.
2. `cd ~/development/rtems/src`
3. `mkdir b-rpi`
4. `cd b-rpi`
5. `../rtems/configure --target=arm-rtems4.11 --enable-rtemsbsp=raspberrypi --enable-tests=samples --enable-networking --enable-posix --prefix=/opt/rtems4.11/rpi`
6. `make`
7. `sudo PATH=$HOME/development/rtems/4.11/bin:$PATH make install`
8. To verify that the build and installation worked correctly, execute the following commands. Results similar to those shown in Figure 10 and Figure 11 **Error! Reference source not found.**, respectively, are expected.
 - a. `ls -artl /opt/rtems-4.11/rpi`
 - b. `find . -name '*.exe' -print`

A terminal window titled "Terminal - user@localhost: ~" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the command `ls -artl /opt/rtems-4.11/rpi` and its output. The output lists files and directories with permissions, owner, group, size, and timestamps. The files are: `.` (7 root root 4096 Nov 14 18:00), `bin` (2 root root 4096 Nov 14 18:40), `lib` (3 root root 4096 Nov 14 18:40), `make` (3 root root 4096 Nov 14 18:41), `share` (4 root root 4096 Nov 14 18:41), `arm-rtems4.11` (3 root root 4096 Nov 14 18:41), and `..` (5 root root 4096 Dec 12 17:19). The prompt `[user@localhost ~]$` is followed by a green cursor.

```
Terminal - user@localhost: ~
File Edit View Terminal Tabs Help
user@localhost:~/Documents x user@localhost:~ x
[user@localhost ~]$ ls -artl /opt/rtems-4.11/rpi
total 28
drwxr-xr-x. 7 root root 4096 Nov 14 18:00 .
drwxr-xr-x. 2 root root 4096 Nov 14 18:40 bin
drwxr-xr-x. 3 root root 4096 Nov 14 18:40 lib
drwxr-xr-x. 3 root root 4096 Nov 14 18:41 make
drwxr-xr-x. 4 root root 4096 Nov 14 18:41 share
drwxr-xr-x. 3 root root 4096 Nov 14 18:41 arm-rtems4.11
drwxr-xr-x. 5 root root 4096 Dec 12 17:19 ..
[user@localhost ~]$
```

Figure 10. Installed Raspberry Pi tools.

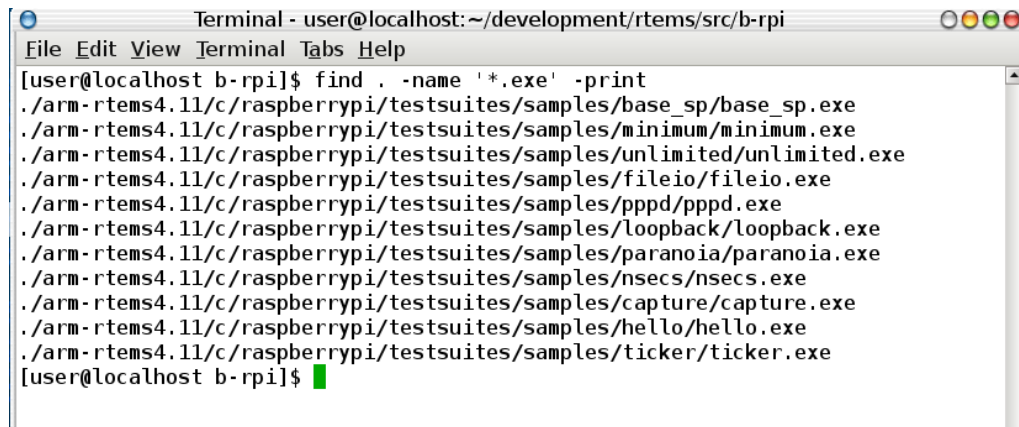
A terminal window titled "Terminal - user@localhost: ~/development/rtems/src/b-rpi". The window shows the output of the command `find . -name '*.exe' -print`. The output lists ten sample executables located in the directory `./arm-rtems4.11/c/raspberrypi/testsuites/samples/`. The executables are: `base_sp/base_sp.exe`, `minimum/minimum.exe`, `unlimited/unlimited.exe`, `fileio/fileio.exe`, `pppd/pppd.exe`, `loopback/loopback.exe`, `paranoia/paranoia.exe`, `nsecs/nsecs.exe`, `capture/capture.exe`, and `hello/hello.exe`. The terminal prompt is `[user@localhost b-rpi]$` with a green cursor.

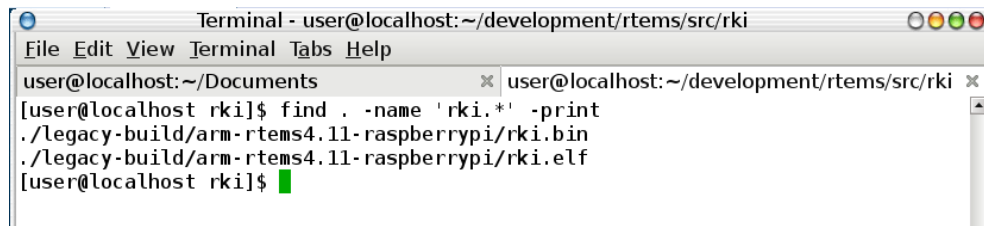
Figure 11. Raspberry Pi sample executables.

Step 3: Build the RKI executable

The RTEMS Kernel Image (RKI) executable that is found on the *RTEMS on Raspberry Pi* blog [17] allows more detailed testing of an RTEMS executable on the Raspberry Pi. The steps to download and build this executable are:

1. Login as *user* and execute the following commands.
2. `cd ~/development/rtems/src`
3. `git clone http://github.com/alanc98/rki.git`
4. `cd rki`
5. The file *Makefile* included in the downloaded source is specific to the blog owner's environment and must be modified to work in the environment described herein. To do this, execute the following commands:
 - a. `mv Makefile Makefile.orig`
 - b. `cp Makefile.orig Makefile`
 - c. Edit the file *Makefile*, make the following changes:
 - Change the line `RTEMS_TOOL_BASE ?= /home/alan/Projects/rtems/4.11` to `RTEMS_TOOL_BASE ?= /home/user/development/rtems/4.11`.
 - Change the line `RTEMS_BSP_BASE ?= /home/alan/Projects/rtems/4.11` to `# RTEMS_BSP_BASE ?= /home/alan/Projects/rtems/4.11`.
 - Change the line `WARNINGS = -Wall` to `WARNINGS = -Wall -Wno-unused-but-set-variable`.

6. To build the RKI executable, execute `'make ARCH=arm-rtems4.11 BSP=raspberrypi RTEMS_BSP_BASE=/home/user/development/rtems/src/b-rpi'`.
7. To verify that the build was successful, execute `'find . -name 'rki.*' -print'`. A result similar to that shown in Figure 12 is expected.



```
Terminal - user@localhost: ~/development/rtems/src/rki
File Edit View Terminal Tabs Help
user@localhost: ~/Documents x user@localhost: ~/development/rtems/src/rki x
[user@localhost rki]$ find . -name 'rki.*' -print
./legacy-build/arm-rtems4.11-raspberrypi/rki.bin
./legacy-build/arm-rtems4.11-raspberrypi/rki.elf
[user@localhost rki]$
```

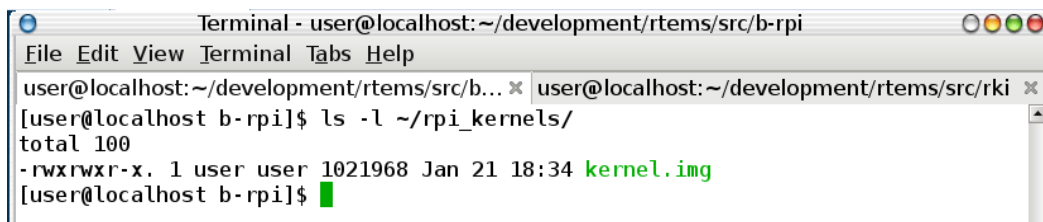
Figure 12. RKI build status.

Step 4: Create a bootable application

Before a sample application can run on the Raspberry Pi, its executable must be converted into a file type that the Raspberry Pi boot loader recognizes. The steps to convert the *ticker* executable are:

1. Login as *user* and execute the following commands.
2. `cd ~/development/rtems/src/b-rpi`
3. `mkdir ~/rpi_kernels`
4. `arm-rtems4.11-objcopy -Obinary arm-rtems4.11/c/raspberrypi/testsuites/samples/ticker/ticker.exe ~/rpi_kernels/kernel.img`
5. To verify that the file was converted, execute `'ls -l ~/rpi_kernels'`.

A result similar to that shown in Figure 13 is expected.



```
Terminal - user@localhost: ~/development/rtems/src/b-rpi
File Edit View Terminal Tabs Help
user@localhost: ~/development/rtems/src/b-rpi x user@localhost: ~/development/rtems/src/rki x
[user@localhost b-rpi]$ ls -l ~/rpi_kernels/
total 100
-rwxrwxr-x. 1 user user 1021968 Jan 21 18:34 kernel.img
[user@localhost b-rpi]$
```

Figure 13. A Raspberry Pi bootable file

Step 5: Copy the converted RTEMS executable to a Windows 7 system

Transfer the bootable file from the development system to the Windows 7 system.

The steps to do this are:

1. On the development system, login as *user* and execute the following commands.
2. `sudo mkdir /mnt/cifs`
This command creates a mount point for the Windows share.
3. `sudo mount -t cifs -o username=<Windows user>,domain=<Windows domain> //<Windows IP>/<Windows share> /mnt/cifs`[24]. Where
 - <Windows user> is replaced by the username on the Windows system.
 - <Windows domain> is replaced by the domain of the Windows system.
 - <Windows IP> is replaced by the IP address, or name, of the Windows system.
 - <Windows share> is replaced by the name of the share on the Windows system.This command mounts the Windows share.
4. `sudo cp /home/user/development/rpi_kernels/kernel.img /mnt/cifs/<path>`. Where
 - <path> is replaced by the appropriate path on the Windows share.This command copies the file to the Windows share.
5. `sudo umount /mnt/cifs`
This command unmounts (disconnects) the Windows share.

Step 6: Prepare a bootable SD card

To boot the Raspberry Pi, a micro SD card is used. The steps to copy the boot files and application from the Windows 7 system to the SD card are:

1. Ensure that the boot files listed in Section II.C.6 have been downloaded to the Windows 7 system.
2. Insert the SD card reader into a USB port on the Windows 7 system.
3. If the SD card has any previous contents, save the files if necessary, and delete all files from the SD card.
4. Copy the boot files to the root (top level) folder of the SD card.
5. Copy the converted RTEMS executable file (created in Step 4 above) to the root folder of the SD card.

The contents of the SD card should be similar to that shown in

6. Figure 14.

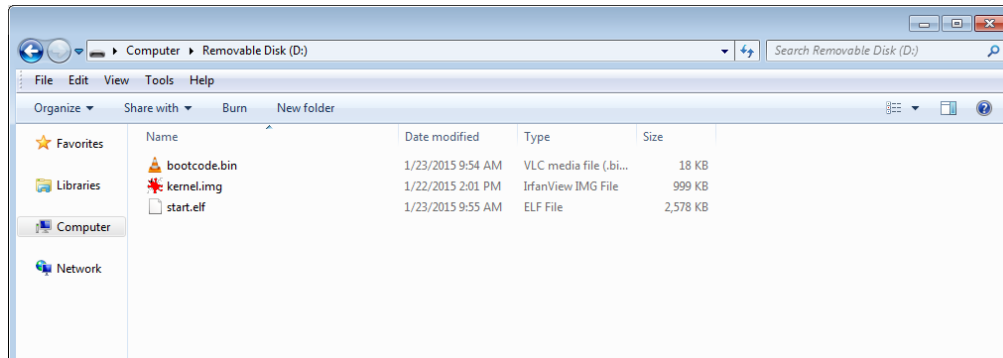


Figure 14. Contents of SD card ready for booting on the Raspberry Pi

Step 6: Run the application

The steps to run the *ticker* application on the Raspberry Pi are:

1. Ensure the Raspberry Pi is powered off.
2. Remove the existing SD card, if any.
3. Install the SD card prepared in the previous step.
4. Attach a USB-serial cable [20] to the Raspberry Pi according to the instructions in [21]. Do NOT use the red power cable.
5. Attach the USB-serial cable to the console computer. The instructions in [21] include details showing how to install and run a terminal emulator on various operating systems.
6. Start the terminal emulator program.
7. Apply power to the Raspberry Pi.
8. The output should be similar to that shown in Figure 15.

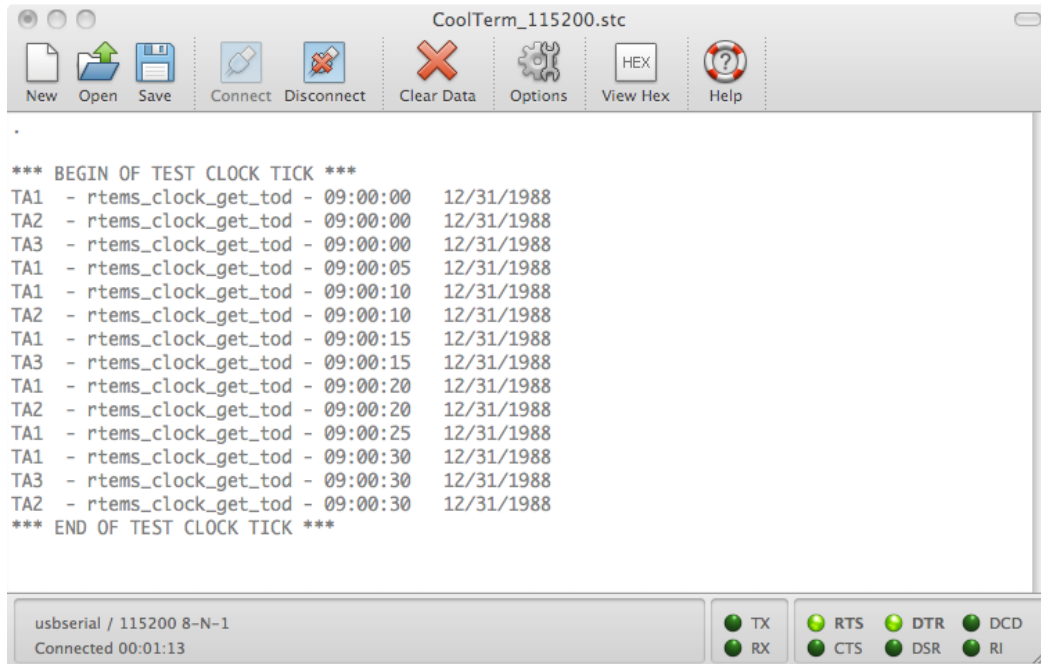
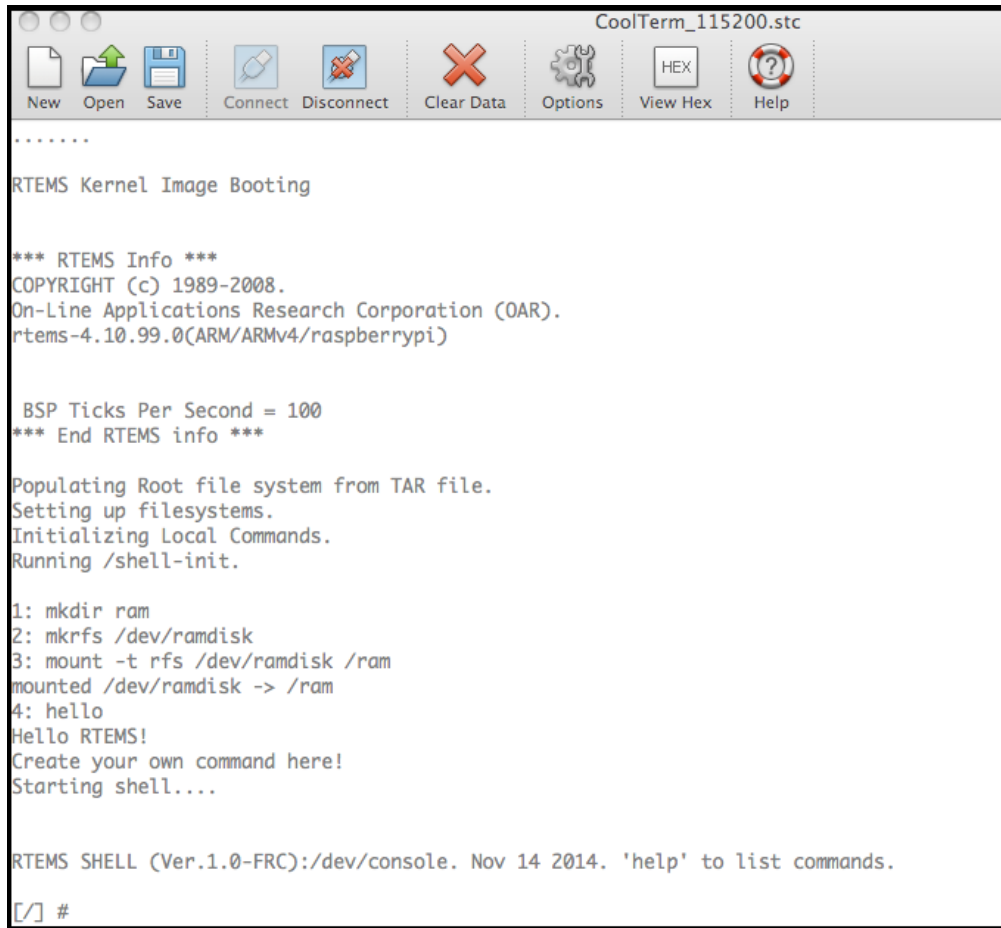


Figure 15. Output of ticker application on Raspberry Pi

Step 7: Run the RKI on the Raspberry Pi

To run the RKI on the Raspberry Pi, use the Windows system to prepare the SD card with the boot files, copy the RKI to the Windows system, then to the SD card, then boot the Raspberry Pi. The steps to do this are:

1. If the SD card has any previous contents, save the files if necessary, and delete all files from the SD card.
2. Copy the boot files (see II.C.6) to the root (top level) folder of the SD card.
3. Unlike the ‘ticker’ example above, the RKI binary file does not need any conversion. The file to copy is located on the development system at */home/user/development/rtems/src/b-rpi/legacy-build/arm-rtems4.11-raspberrypi/rki.bin*. Copy this file to the Windows systems using Step 5 above as an example. Copy this file to the root folder of the SD card.
4. Rename the file on the SD card to *kernel.img*.
5. Follow Step 6 above to run the RKI application. No modifications to this procedure are required. The initial screen should be similar to that shown in Figure 16.



```
.....
RTEMS Kernel Image Booting

*** RTEMS Info ***
COPYRIGHT (c) 1989-2008.
On-Line Applications Research Corporation (OAR).
rtems-4.10.99.0(ARM/ARMv4/raspberrypi)

BSP Ticks Per Second = 100
*** End RTEMS info ***

Populating Root file system from TAR file.
Setting up filesystems.
Initializing Local Commands.
Running /shell-init.

1: mkdir ram
2: mkrfs /dev/ramdisk
3: mount -t rfs /dev/ramdisk /ram
mounted /dev/ramdisk -> /ram
4: hello
Hello RTEMS!
Create your own command here!
Starting shell....

RTEMS SHELL (Ver.1.0-FRC):/dev/console. Nov 14 2014. 'help' to list commands.

[/] #
```

Figure 16. RKI start-up screen

Warning: The RKI application executes with *root* privileges. Care should be taken to not delete any system files, or make changes that could cause the system to become unusable.

The startup of the RKI automatically executes the commands contained in the file */shell-init* (see Figure 16). This file includes commands to create and mount a RAM disk, which allows direct access to a portion of the Raspberry Pi RAM as a file system. Following the initialization, an RTEMS shell is started to allow the user to execute arbitrary commands. There are commands to manipulate files, system date, time and real-time clock, tasks, and many more. A list of commands can be seen by using the help command. Full documentation of the RTEMS Shell can be found in [22]. Figure 17 is an example of the help screen for file-related commands.

```

CoolTerm_115200.stc
New Open Save Connect Disconnect Clear Data Options View Hex Help

[/] # help files
help: list for the topic 'files'
cp      - cp [-R [-H | -L | -P]] [-f | -i] [-pv] src target
pwd     - pwd # print work directory
ls      - ls [dir] # list files in the directory
chdir   - chdir [dir] # change the current directory
mkdir   - mkdir dir # make a directory
rmdir   - rmdir dir # remove directory
chroot  - chroot [dir] # change the root directory
chmod   - chmod 0777 n1 n2... # change filemode
cat     - cat n1 [n2 [n3...]] # show the ascii contents
mkrfs   - mkrfs [-v] [-s blksize] [-b grpblk] [-i grpinode] [-I] [-o %inode]
dev
mkdos   - mkdos [-V label] [-s sectors/cluster] [-r size] [-v] path # forma
t disk
msdosfmt - is an <alias> for command 'mkdos'
mv      - [-fiv] source target ...
rm      - [-f | -i] [-dIPRrvw] file ...
Press any key to continue...
ln      - ln ln [-fhinsv] source_file [target_file]
mknod   - mknod mknod [-rR] [-F fmt] [-m mode] name [c | b] minor
lsof    - lsof
mount   - mount [-t type] [-r] [-L] [-o options] source target
unmount - unmount path # unmount disk
blksync - blksync driver # sync the block driver
blkstats - blkstats [-rl--reset] PATH_TO_DEVICE
fdisk   - disk format and utility functions

fdisk DISK_NAME
prints the partition table

fdisk DISK_NAME [FS N1 [N2 ... ]] ... [write] [FORMAT]
creates a new partition table

fdisk DISK_NAME register
creates a logical disk for each partition of the disk

fdisk DISK_NAME unregister
deletes the logical disks associated with the partitions of the di
sk

fdisk DISK_NAME mount
mounts the file system of each partition of the disk

fdisk DISK_NAME unmount
unmounts the file system of each partition of the disk

option values:

usbserial / 115200 8-N-1
Connected 00:03:42
TX RX RTS CTS DTR DSR DCD RI

```

Figure 17. RKI file-related commands

There are also two benchmark-related commands *dhrystone*, and *whetstone*, which can be used to compare RTEMS performance against other real-time operating systems. Figure 18 is an example of the output of the *whetstone* benchmark.

```

[/] #
[/] # whetstone
Running Whetstone Command!
  0      0      0  1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00
  60     70     60 -6.7500e-02 -4.6465e-01 -7.3153e-01 -1.1286e+00
  70     60     60 -6.0913e-02 -4.5692e-01 -7.2207e-01 -1.1181e+00
 1725    0      0  1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00
 1050    1      2  6.0000e+00  6.0000e+00 -7.2207e-01 -1.1181e+00
 160     1      2  4.9518e-01  4.9518e-01  4.9516e-01  4.9516e-01
 4495    1      2  1.0000e+00  1.0000e+00  9.9994e-01  9.9994e-01
 3080    1      2  3.0000e+00  2.0000e+00  3.0000e+00 -1.1181e+00
   0      2      3  1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00
  465    2      3  7.9611e-01  7.9611e-01  7.9611e-01  7.9611e-01

Loops: 5, Iterations: 100, Duration: 7 sec.
C Converted Double Precision Whetstones: 7.1 MIPS
Completed Whetstone

[/] #
[/] #

```

Figure 18. RKI whetstone results

Additionally, there are commands that display information about RTEMS resource usage, see Figure 19.

```

[/] # cpuuse
-----
CPU USAGE BY THREAD
-----
ID      | NAME      | SECONDS | PERCENT
-----
0x09010001 | IDLE      | 1726.080000 | 98.822
0x0a010001 | UI1       | 0.590000 | 0.033
0x0a010002 | BSWP      | 0.000000 | 0.000
0x0a010004 | shel      | 19.810000 | 1.134
-----
TIME SINCE LAST CPU USAGE RESET IN SECONDS: 1746.670000
-----

[/] #
[/] # stackuse
Stack usage by thread
ID      NAME      LOW      HIGH      CURRENT  AVAILABLE  USED
-----
0x09010001 IDLE 0x00159608 - 0x0015a607 0x0015a598 4080      136
0x0a010001 UI1  0x0015a610 - 0x0016260f 0x001624f8 32752     804
0x0a010002 BSWP 0x00162618 - 0x00163617 0x00163560 4080      276
0x0a010004 shel 0x00163620 - 0x0016861f 0x00168058 20464     3960

[/] #
[/] # perioduse
Period information by period
--- CPU times are in seconds ---
--- Wall times are in seconds ---
ID      OWNER COUNT MISSED      CPU TIME      WALL TIME
      MIN/MAX/AVG      MIN/MAX/AVG

[/] #
[/] # wkspace
C Program Heap and RTEMS Workspace are separate.
Number of free blocks: 1
Largest free block: 715552
Total bytes free: 715552
Number of used blocks: 33
Largest used block: 175448
Total bytes used: 333032

[/] #

```

Figure 19. RKI resource usage

IV. CONCLUSION AND FUTURE WORK

This report describes the results of our experiments with RTEMS. Our motivation was to gain an understanding of the inner working of RTEMS to support research on network-based covert communications in space systems constructed with multilevel security capabilities. We have provided a set of instructions on how to build and run RTEMS in a SPARC simulator and on a Raspberry Pi computer.

Future work includes extending the RKI application to handle Ethernet networking, and running the Core Flight Software on RTEMS executing on the Raspberry Pi. These two efforts will help further our understanding of both RTEMS internals and the Core Flight Software system.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- [1] On-line Applications Research Corporation, "RTEMS C User's Guide, edition 4.10.2 for RTEMS 4.10.2," December 2011.
- [2] Aeroflex Gaisler AB, "Operating Systems Compilers and Real-time Operating Systems for LEON and ERC32," April 2010. Available at: http://www.gaisler.com/doc/operating_systems_product_sheet.pdf. Accessed: January 2015.
- [3] On-line Applications Research Corporation, "RTEMS Wiki." Available at: <https://devel.rtems.org/wiki/TBR/BSP/Rad750>. Accessed: January 2015.
- [4] Saab Ericsson Space, "Spacecraft Management Unit," March 2005. Available at: https://www.rtems.org/sites/default/files/Spacecraft_Management_Unit_Saab-2011-10.pdf. Accessed: January 2015.
- [5] On-line Applications Research Corporation, "RTEMS Application Spotlight Electra," October 2011. Available at: <https://www.rtems.org/sites/default/files/Application-Electra-2011-10.pdf>. Accessed: January 2015.
- [6] National Aeronautics and Space Administration, "NASA's Core Flight Software - a Reusable Real-Time Framework," November 2014. Available at: <http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20140017040.pdf>. Accessed: January 2015.
- [7] Nguyen, T. D., "A Study of Covert Communications in Space Platforms Hosting Government Payloads," Naval Postgraduate School Technical Report NPS-CAG-15-002, January 2015. (To be published)
- [8] On-line Applications Research Corporation, "RTEMS." Available at: <https://www.rtems.org/>. Accessed: January 2015.
- [9] On-line Applications Research Corporation, "RTEMS CPUs." Available at: <https://devel.rtems.org/wiki/TBR/UserManual/SupportedCPUs>. Accessed: January 2015.
- [10] On-line Applications Research Corporation, "RTEMS Board Support Packages." Available at: https://devel.rtems.org/wiki/TBR/Website/Board_Support_Packages. Accessed: January 2015.
- [11] Military Standard MIL-STD-1553B: "Aircraft Internal Time Division Command/Response Multiplex Data Bus," September 21, 1978.
- [12] European Cooperation for Space Standardization, "ECSS-E-ST-50-12C SpaceWire - Links, nodes, routers and networks," July 2008.
- [13] BAE Systems, "BAE RAD750." Available at: <http://www.baesystems.com/our-company-rzz/our-businesses/electronic-systems/product-sites/space-products-and-processing/processors>. Accessed: January 2015.
- [14] Aeroflex Gaisler AB, "GR-LEON4-ITX LEON4." Available at: <http://gaisler.com/index.php/products/boards/gr-leon4-itx>. Accessed: January 2015.

- [15] Aeroflex Gaisler AB, "GR-CPCI-LEON4-N2X." Available at: <http://gaisler.com/index.php/products/boards/gr-cpci-leon4-n2x>. Accessed: January 2015.
- [16] Raspberry Pi. Available at: <http://www.raspberrypi.org/>. Accessed: January 2015.
- [17] Alan C., "RTEMS on Raspberry Pi," March 28, 2013. [blog entry]. Available at: <http://alanstechnotes.blogspot.com/2013/03/rtems-on-raspberry-pi.html>. Accessed: January 2015.
- [18] On-line Applications Research Corporation, "RTEMS Source Builder." Available at: <http://ftp.rtems.org/pub/rtems/people/chrisj/source-builder/source-builder.html>. Accessed: January 2015.
- [19] On-line Applications Research Corporation, "RTEMS Quick Start Guide." Available at: https://devel.rtems.org/wiki/TBR/UserManual/Quick_Start. Accessed: January 2015.
- [20] Adafruit, Raspberry Pi console cable, USB-Serial cable. Available at: <http://www.adafruit.com/product/954>. Accessed: January 2015.
- [21] Adafruit, "Using the Raspberry Pi console cable." Available at: <https://learn.adafruit.com/adafruits-raspberry-pi-lesson-5-using-a-console-cable>. Accessed: January 2015.
- [22] On-line Applications Research Corporation, "RTEMS Shell User's Guide." Available at: <http://docs.rtems.org/doc-current/share/rtems/html/shell/index.html>. Accessed: January 2015.
- [23] Red Hat, Inc., "Fedora 19 Installation Guide." Available at: http://docs.fedoraproject.org/en-US/Fedora/19/html/Installation_Guide. Accessed: January 2015.
- [24] Red Hat, Inc., "Mounting Windows shares." Available at: <https://access.redhat.com/solutions/448263>. Accessed: January 2015.
- [25] Motorola, Inc., "MCP750 CompactPCI Single Board Computer Installation and Use," July 2002. Available at: https://www.slac.stanford.edu/exp/glast/flight/docs/MCP750/MCP750_Install.pdf. Accessed: February 2015.
- [26] Simon Tatham, "PuTTY: A Free Telnet/SSH Client." Available at: <http://www.chiark.greenend.org.uk/~sgtatham/putty/>. Accessed: February 2015.
- [27] Roger Meier, "Roger Meier's Freeware." Available at: <http://freeware.the-meiers.org/>. Accessed: February 2015.
- [28] Todd C. Miller, "Visudo Manual." Available at: <http://www.sudo.ws/visudo.man.html>. Accessed: February 2015.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California
3. Research Sponsored Programs Office, Code 41
Naval Postgraduate School
Monterey, CA 93943